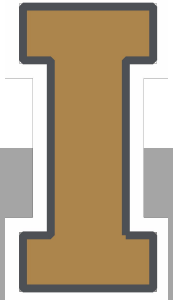


# Team MusIQ

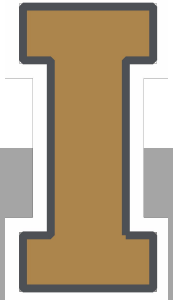
Snare Drum Notator

Scott Dennis, Nathan Groggett,  
Phillip Kearns, Hue Purkett,  
Domn Werner



# Goal

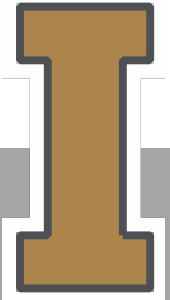
- **Facilitate the creation of snare drum sheet music.**
- **Allow you to write music without knowing how.**



# Requirements - Hardware

## **The product shall:**

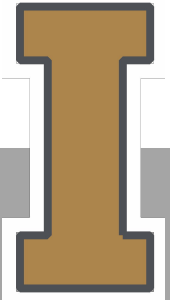
- Be worn comfortably on the top of both of the user's hands and not interfere with the user's playing ability.
- Capture and communicate movement data wirelessly.
- Be rechargeable.



# Requirements - Software

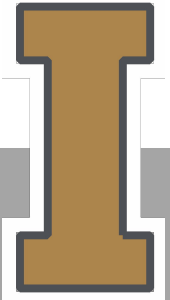
## **A medium of interaction that will:**

- Allow the user to input a tempo and time signature.
- Allow the user to start and stop the recording process.
- Produce accurate and complete snare drum sheet music in MusicXML format.



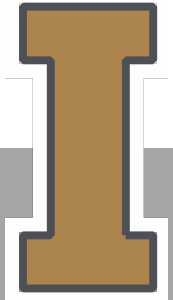
# Specifications

- Recording
  - Record rhythm, volume, sticking, and rudiments that the user plays.
- Display
  - Saved file of the notes and sheet music played in MusicXML.
- User Input
  - User shall be able to set the tempo.
  - User shall be able to start and stop the data capture.



# Constraints

- Support between 60 and 240 beats per minute.
- Support up to 16 notes per measure.
- Support from quarter note to 16th note subdivisions.
- Sensors should not reduce the drummer's ability to play.



# Edison

## Key Features

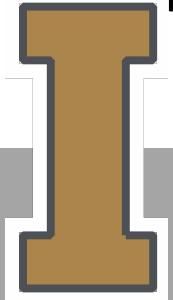
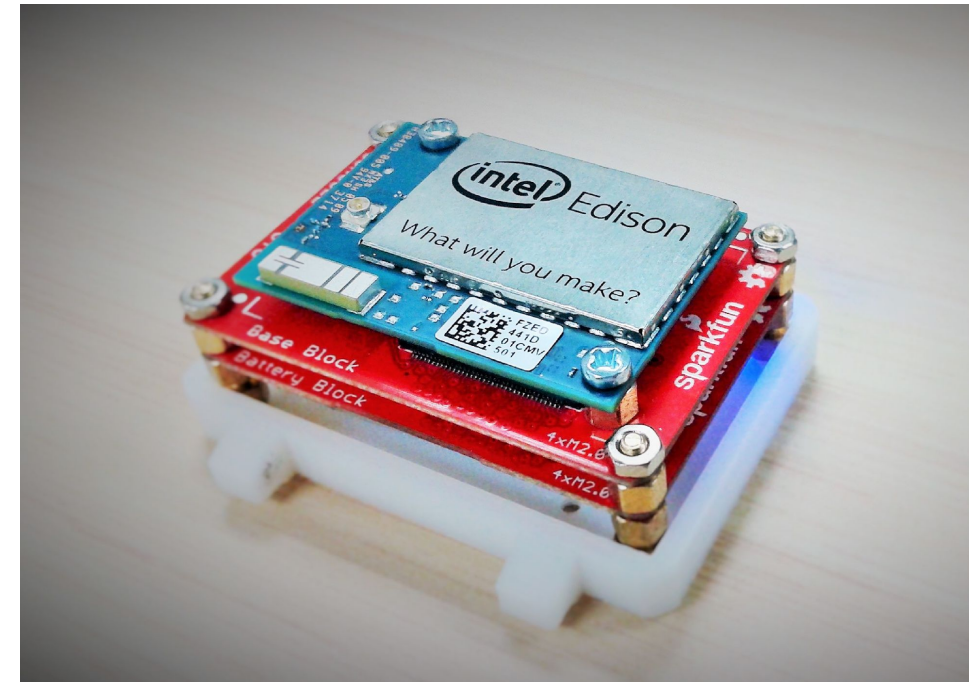
- Wi-Fi, Bluetooth LE
- Multiple development environments (Arduino, Python)
- Wide array of breakout add-on boards - 100 Hz data

## Specs

- Intel Atom SoC, dual-core CPU @ 500 MHz
- 1 GB RAM, 4 GB Flash

## Uses

- Hand-mounted sensor
  - Strike detection/data logging, motion classification



# Additions - SparkFun Blocks

## Base Block

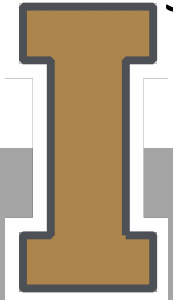
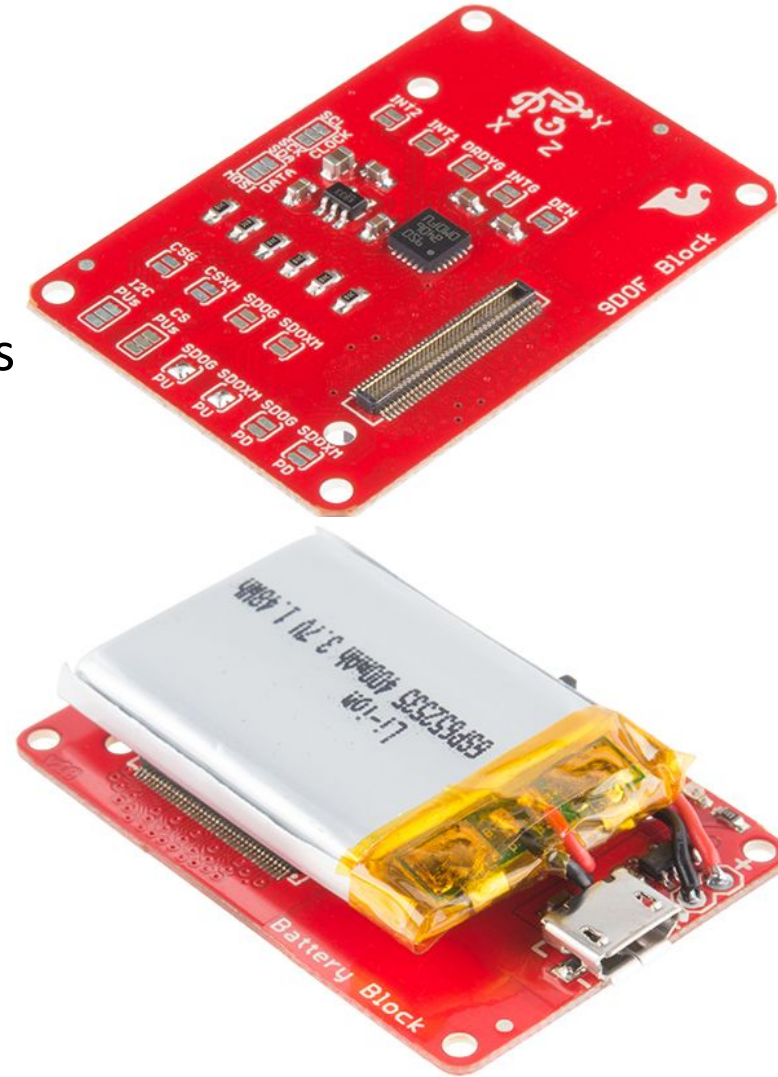
- Micro AB USB ports allows for attachment of various peripherals
  - Console (programming), OTG (storage)

## 9DOF Block

- 3-axis accel/gyro/mag + temperature through I2C
  - Standard (100kHz) and Fast (400kHz) modes
  - Adjustable ranges for each sensor ( $\pm 2 - 16$  G, etc.)

## Battery Block

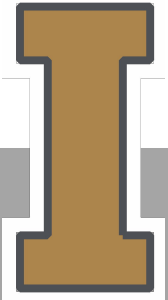
- Single cell LiPo battery
  - 400mAh ~ 4-8 hours continual use





# Cost Estimates - Cheap!

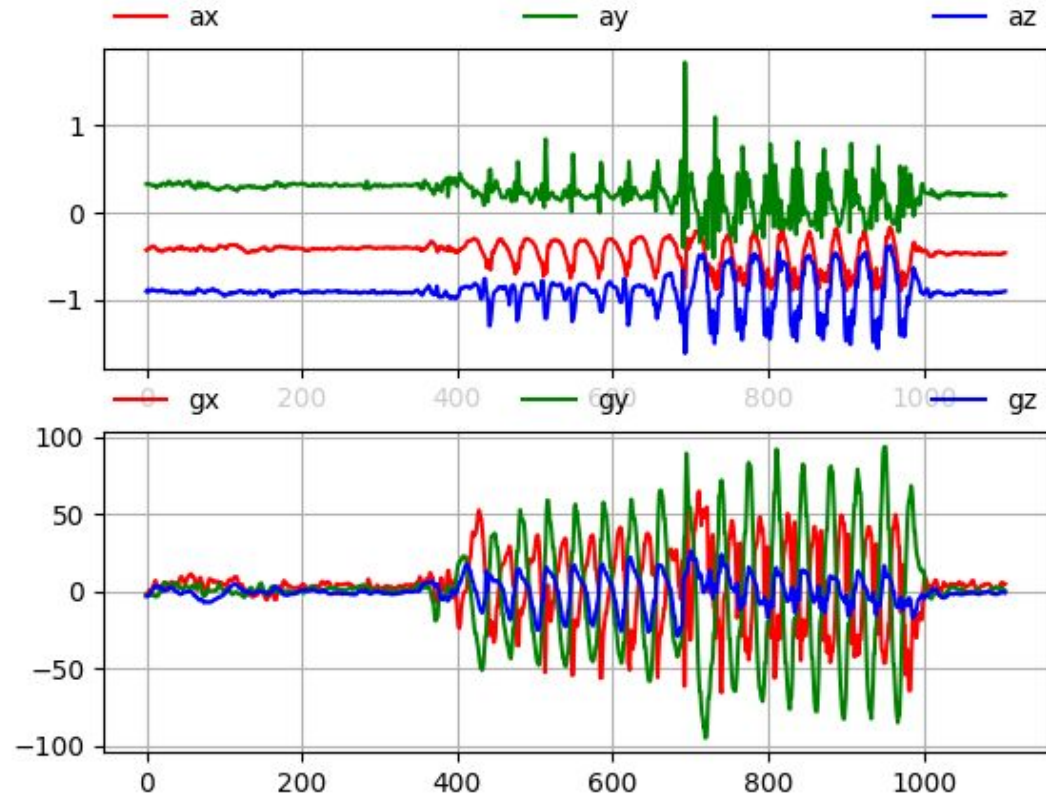
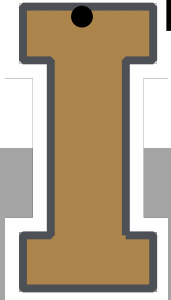
Product	Quantity	Price
Intel Edison	2	\$100
Battery, Base, 9DOF	2 (6)	\$185
Gloves/Velcro	2	\$20
Hardware	2	\$6
<b>TOTAL</b>		<b>\$311</b>



# Hit Detection - Accel and Gyro

## Sensors on Hands

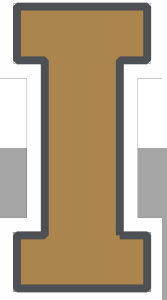
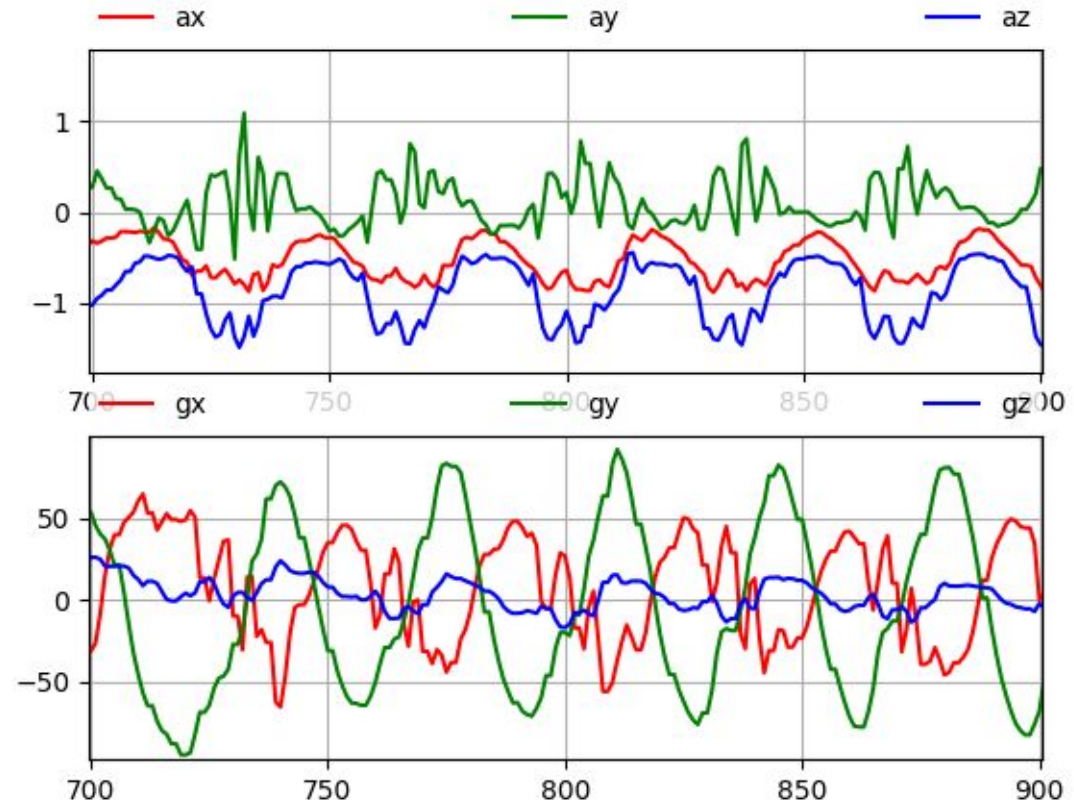
- Excellent resolution
  - See each part of the motion
- Limited latency
  - Short time to detect a peak/note
- **Which arm**
- Intensity
- More data = more precision



# Hit Detection - Accel and Gyro

## Which Data to Use

- Gyroscope
  - Clearer, smoother peaks
  - Determine **when** a note occurs
- Accelerometer
  - Greater sensitivity
  - Determine **what** note occurs



# Language/Framework Choices

## Edison Sensor Reading and Output

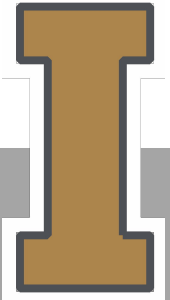
- Python
- Modified existing SPP Loopback file from Intel.

## Reading Output and Parsing

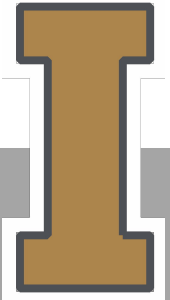
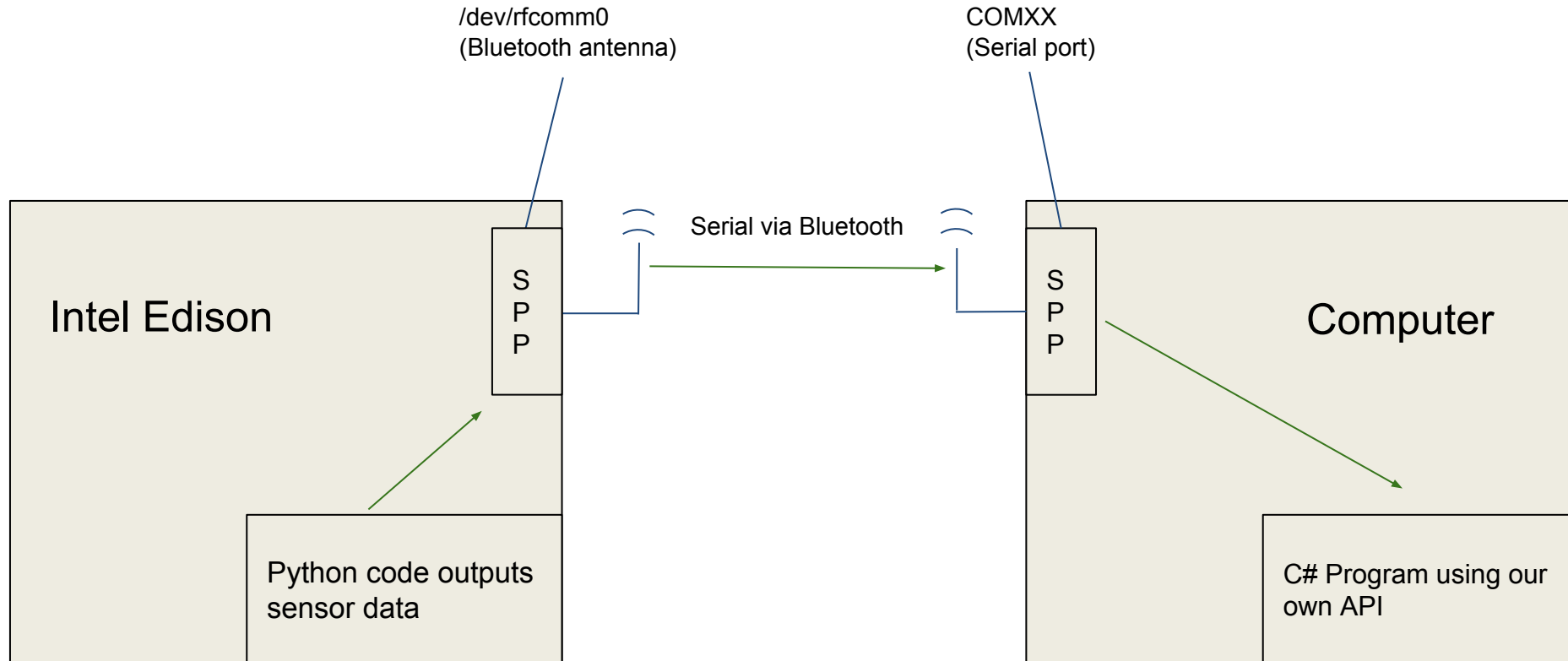
- C#
- Wrote our own API as a portable library.

## Graphical User Interface (GUI)

- C# for code-behind.
- Windows Forms for UI elements.



# Bluetooth Communication Diagram between Intel Edison and Computer



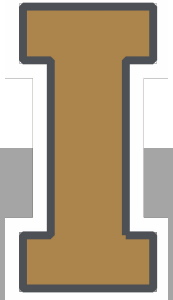
# Bluetooth

RFcomm communication protocol

- creates virtual serial ports for communication
- 2 for input, 2 for output

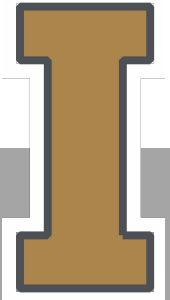
Communication libraries

- pyBlueZ
- 32feet C# library by in the hand




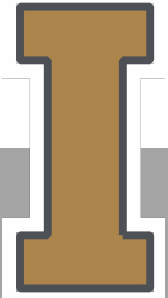
# Python Code On Edison

- Identical Code on each Edison
  - Modified from SPP Loopback from Intel
- Initializes the 9DOF IMU
  - Set sampling rates, sensitivity ranges
  - Calibrate for zero offset and zero time
- Creates LSM9DS0 class, imu object
  - Wait for new data from sensors
  - Update variables on object
- Broadcasts out new data via SPP for bluetooth
- Waits for stop gesture.




# Music Xml Generation

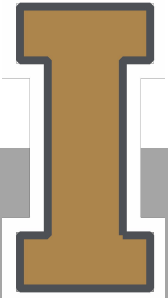
Sheet Music	Function Calls	MusicXml Output
	<pre>MusicXmlGenerator g = new MusicXmlGenerator(); g.createMeasure();</pre>	<pre>&lt;measure number="1"&gt;   &lt;attributes&gt;     &lt;time&gt;       &lt;beats&gt;4&lt;/beats&gt;       &lt;beat-type&gt;4&lt;/beat-type&gt;     &lt;/time&gt;     &lt;clef&gt;       &lt;sign&gt;percussion&lt;/sign&gt;       &lt;line&gt;2&lt;/line&gt;     &lt;/clef&gt;   &lt;/attributes&gt; &lt;/measure&gt;</pre>






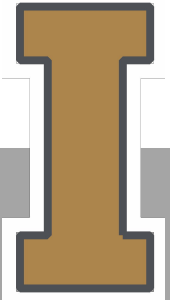
# Music Xml Generation

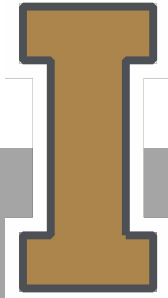
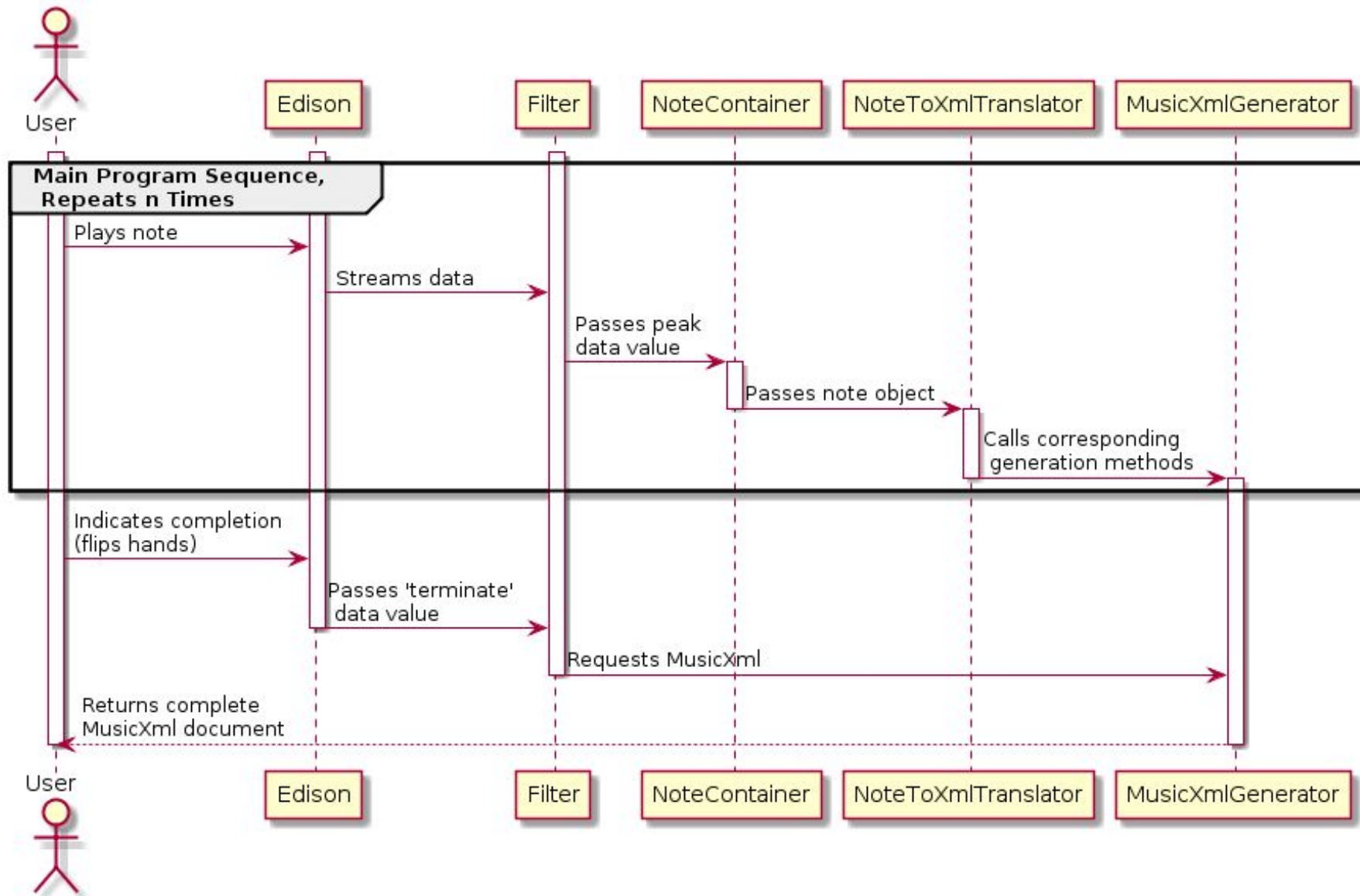
Sheet Music	Function Calls	MusicXml Output
	<pre>MusicXmlGenerator g = new MusicXmlGenerator(); g.Add(new Rhythm("quarter"));</pre>	<pre>&lt;note&gt;   &lt;unpitched&gt;     &lt;display-step&gt;C&lt;/display-step&gt;     &lt;display-octave&gt;5&lt;/display-octave&gt;   &lt;/unpitched&gt;   &lt;type&gt;quarter&lt;/type&gt;   &lt;stem&gt;up&lt;/stem&gt;   &lt;duration&gt;1&lt;/duration&gt; &lt;/note&gt;</pre>



# Music Xml Generation

Sheet Music	Function Calls	Music Xml Output
	<pre>MusicXmlGenerator g = new MusicXmlGenerator(); g.Add(new BuzzStrokeRoll(new Accent(new Rhythm("quarter"))));</pre>	<pre>&lt;note&gt;   &lt;unpitched&gt;     &lt;display-step&gt;C&lt;/display-step&gt;     &lt;display-octave&gt;5&lt;/display-octave&gt;   &lt;/unpitched&gt;   &lt;type&gt;quarter&lt;/type&gt;   &lt;stem&gt;up&lt;/stem&gt;   &lt;duration&gt;1&lt;/duration&gt;   &lt;notations&gt;     &lt;articulations&gt;       &lt;accent /&gt;     &lt;/articulations&gt;     &lt;ornaments&gt;       &lt;tremolo type="single"&gt;3&lt;/tremolo&gt;     &lt;/ornaments&gt;   &lt;/notations&gt; &lt;/note&gt;</pre>





# Questions?

